Estimation de paramètres ; nouveautés après ICTAI 15

Bertrand Neveu

August 11, 2016

Voici les principaux changements par rapport à la version de l'article d'ICTAI

1 Condition d'arrêt

Arrêt d'une branche dans version ICTAI Condition d'arrêt pour la branche courante (pas de bissection) dans ICTAI : précision atteinte ou possibles = valides. La gestion des solutions partageant des observations valides a lieu dans le post-traitement et on ne garde que les solutions maximales, ne partageant pas plus de la moitié de leurs observations avec une autre solution maximale.(cf partie 8)

Nouvelle condition d'arrêt Nouvelle condition : on s'arrête quand il ne reste pas plus de Q/2 points possibles non validés : toute autre solution dans la boîte courante partagerait plus de la moitié de ses points valides avec la solution trouvée.

Avantage : on s'arrête plus tôt dans l'arbre de recherche, surtout quand il y a des plans avec des nombres de mesures différentes.

Inconvénient : on n'a plus les solutions optimales en ensemble de points valides et on ne sait plus caractériser les solutions rendues par le post-traitement (cf partie 8), cet ensemble de solutions après post-traitement dépendant alors de l'ordre dans lequel ces solutions ont été trouvées.

La seule chose que l'on sait est que les solutions non recherchées sont proches des solutions trouvées, dans le sens qu'elles auraient partagé avant posttraitement la moitié de leurs mesures avec une solution trouvée.

Cette nouvelle condition est actuellement optionnelle.

2 Recherche d'outliers

cf mail du Chili 11/12

La recherche est lancée si possibles -Q est inférieur à un seuil.

La recherche des ouliers s'effectue en prenant des triplets d'observations disjoints et en cherchant si la solution correspondante à un triplet donné a une intersection vide avec la boîte courante, auquel cas on a trouvé qu'il y a un

outlier dans ce triplet (mais on ne sait pas de quelle observation il s'agit). On rejette la boîte courante si on a trouvé plus d'outliers que la différence entre le nombre de mesures possibles et Q. Ce seuil est un paramètre de l'algorithme : dans les exemples, on a essayé de le mettre entre 0 et 20. La diminution du nombre de boites assez sensible (20 %), mais le temps de résolution varie peu. L'algo ICTAI est obtenu en mettant ce paramètre seuil = -1. (on n'effectue aucun test).

3 Recherche de points valides

3.1 Limitation recherche point valide

Par ailleurs, on a limité le lancement de la recherche du point valide en accord avec la condition d'arrêt: on ne lance cette recherche que si on a une possibilité d'obtenir la condition d'arrêt.

3.2 Mini ransac

cf mail du Chili 11/12 Au lieu de rechercher à valider les observations sur le point milieu de la boîte courante, on essaye à partir de k triplets tirés aléatoirement. k, le nombre de triplets est un paramètre de l'algorithme.

- point milieu (k=0) : calcul des mesures valides au point milieu de la boîte courante (algo ICTAI)
- k : chaque point pour la validation des observations est le milieu de l'intersection de la boite courante avec la solution de l'intersection de 3 mesures choisies aléatoirement (mini-ransac)

Les meilleurs résultats sont obtenus avec k =1 , mais ils varient assez peu pour $k <= 3$. Au delà, cette recherche est trop chère.

3.3 Limitation recherche point valide

Par ailleurs, on a limité le lancement de la recherche du point valide en accord avec la condition d'arrêt: on ne lance cette recherche que si on a une possibilité d'obtenir la condition d'arrêt.

4 Limitation des q-intersections

(cf mail 24/12) Quand on choisit de faire des q-intersections dans la direction supplémentaire, un seuil est mis pour le lancement de la q-intersection sur les directions de la boite courante. Ce seuil est determiné par : possibles - Q. Dans la version courante, le seuil est fixé à 10 pour les plans, c.à.d les q-intersections dans les directions de la boîte courante ne sont faites qu'en bas de l'arbre de recherche quand possibles - Q <= seuil.

5 Optimisation

Noux avons ajouté un algorithme d'optimisation (maximisation) sur le nombre de mesures valides (*inliers*). C'est un algorithme de type Branch and Bound avec plusieurs stratégies possibles :

- une stratégie en profondeur d'abord simple sans choix du meilleur des 2 fils
- une stratégie en profondeur d'abord simple avec choix du meilleur des 2 fils : le critère de choix peut être le plus grand nombre d'observations valides ou la borne sup des observations de la boite
- une stratégie en meilleur d'abord.

La stratégie en meilleur d'abord cause assez rapidement des explosions mémoire dans les essais sur la matrice fondamentale, ainsi que le deuxième exemple de scène réelle avec 3611 points (house44).

Un algo de type IbexOpt (contraction des fils et recherche point faisable avant mise dans le tas) a été implanté.

Pour les plans, cette stratégie en profondeur d'abord avec choix du meilleur fils est la plus rapide, dans l'implantation de type IbexOpt.

Autres stratégies ou autres critères à étudier ?

Bornes Pour chaque boîte, le majorant qmax du nombre d'inliers est calculé par l'algorithme de q-intersection par projections comme le plus grand nombre de possibles sur un intervalle projeté contenant au moins q observations possibles. Ce majorant peut être plus petit que le nombre de possibles dans la boîte courante (en particulier s'il y a plusieurs solutions dans la boîte courante).

Si ce majorant est plus petit que q, la branche est abandonnée.

Le minorant est le nombre de valides. La condition d'arrêt de la branche courante est : $qmax = valid$ ou précision atteinte.

Le critère à optimiser étant un entier, quand on a trouvé une solution avec k inliers, on recherche par la suite une solution avec au moins q=k+1 inliers. (eps-sol de IbexOpt=1).

Pas de post-traitement : on obtient directement la meilleure solution et une valeur de la borne sup possible sur les petites boites (qsb). (équivalent à uplo-of-epsboxes dans Ibexopt). Si cette borne est inérieure ou égale à la valeur de la meilleure solution, l'optimum est prouvé.

Paramètre de l'algorithme : précision de bissection : si elle est assez petite, on n'obtiendra pas de petites boîtes avec de meilleures solutions potentielles et l'optimum obtenu sera prouvé, mais le temps de calcul risque d'être grand.

En cas d'arrêt en limite de temps : on a une borne supérieure sur les boites restantes. (équivalent à uplo de Ibexopt), mais en profondeur d'abord, cette valeur est grande : on ne fait rien pour la descendre. Ce point n'est pas explicité dans le pseudo-code.

Cet algorithme d'optimisation marche bien sur les plans et les cercles : les optimums trouvés sont prouvés sur tous les exemples artificiels et sur les deux scènes réelles.

Tolérance sur l'objectif : au lieu d'une solution entire optimale , on peut rechercher l'optimum avec une tolérance entire epsobj donnée (équivalent de eps_{sol} de Ibexopt).

Limite des branchements : au del d'une certaine profondeur, on ne poursuit qu'une branche, l'autre (si elle existe) est considérée comme une petite bote et n'est pas mise dans le tas: paramtre de l'algo : profondeur max pour 2 branches choix de la branche : actuellement qmax . peut-on faire mieux ?

```
Algorithm QinterOptim (box, observations, q, ε_sol, τ)
    solution ← ∅
    qsb = 0; node ← new Node
    node.box ← box
    node.possibleCS ← observations
    node.validCS ← ∅
    nodeStack ← {node}
    while nodeStack ≠ ∅ do
        node ← pop (nodeStack)
        box ← node.box
        bisect (box, box_1, box_2) /* split the box */
        node_1 ← quasiCopy (node, box_1)
        node_2 ← quasiCopy (node, box_2)
        handleNode(node_1, q, ε_sol, τ , solution, qsb)
        handleNode(node_2, q, ε_sol, τ , solution, qsb)
        if node_1.isLeaf = false and node_2.isLeaf = false then
            push (nodeStack, worst(node_1,node_2))
            push (nodeStack, best(node_1,node_2))
        else
            if node_1.isLeaf = false then
                push (nodeStack,node_1)
            else
                if node_2.isLeaf = false then
                    push (nodeStack,node_2)
    return (solution, qsb)
```

Algorithm 1: Algorithme d'optimisation donnant une solution contenant le nombre maximum d'inliers : variante en profondeur d'abord avec choix du meilleur fils

5.1 Questions ?

Le branch and bound est classique. Les points originaux sont la détermination du majorant de chaque boîte par l'algo de q-intersection, méthode générique

```
Algorithm handleNode (node, q, ϵ_sol, τ , solution, qsb)
    node.isLeaf ← false
    box ← node.box
    contAndQinter (box, τ, q, node.possibleCS, node.maxpossibles )
    if box ≠ ∅ then
        validateBox (box, node, τ, node.possibleCS, node.validCS)
        if |node.validCS| >= q then
            solution ← node
            q ← |node.validCS| + 1
        if width(box) < ϵ_sol or |node.validCS| = node.maxpossibles then
            if node.maxpossibles > |node.validCS| then
                qsb ← max (qsb, node.maxpossibles)
            node.isLeaf ← true
    else
        node.isLeaf ← true
```

Algorithm 2: Traitement d'un noeud : contraction, calcul des valides et test feuille

indépendante de l'application.

et du point faisable par des "mini-ransac" (mais qui n'améliore qu'à la marge la solution avec la validation sur le point milieu).

Les conclusions sur les améliorations sont les mêmes : les seules améliorations vraiment gagnantes sont celles de la section 4 et celles de la section 3.3, qui suppriment des calculs inutiles.

Il faudrait de nouveaux exemples artificiels (plus réalistes ?) et réels pour être convaincu de l'intérêt de l'approche.

Essais sur la matrice fondamentale avec 2 critères (algébrique et Sampson). On a des difficultés dues à la combinatoire (7 ddl): on n'arrive qu'à optimiser sur des petites boites autour d'une solution trouvée par RANSAC : on améliore localement cette solution.

Pour les problèmes avec contraintes, la recherche d'un point faisable se fait en construisant un polytope intérieur (comme Ibexopt). Question : que minimiser sur ce polytope dans un simplexe pour avoir le plus de points valides ??

5.2 Travaux liés

2 articles de ECCV 2014 qui réalisation une maximisation de l'ensemble des inliers par branch and bound en vision.

- rotation et distance focale (dimension 4) : Globally Optimal Inlier Set Maximization With Unknown Rotation and Focal Length
  Jean-Charles Bazin, Yongduek Seo, Richard Hartley, Marc Pollefeys 4DDL
- Matrice essentielle (dimension 5) : Optimal Essential Matrix Estimation via Inlier-Set Maximization

Yang , Li, Jia ECCV 2014

6 Nouveaux exemples artificiels de plans

- points complètement aléatoires (que du bruit) permet de vérifier que l'algorithme a, à Q fixé, un comportement en m^3 , et que la complexité diminue avec Q.
- nouveaux exemples avec 25 plans
  - sans coordonnées entières : (les exemples ICTAI avaient malencontreusement les coordonnées aléatoires entières) plus de petites boites avec stratégie round robin
  - sans plans parallèles ni coordonnées entières : la recherche de points valides sur ces exemples permet de s'affranchir de la précision. il n'y a plus de petites boites (avec stratégie round robin ou spécifique rr2)

7 Triplets de mesures

Résolution avec tous les triplets : complexité m^3 en temps et en mémoire due à la phase de construction des boites pour tous les triplets. Cette phase est constante à m donné (indépendante de Q) et rend cet algorithme peu efficace. Contrairement à l'algorithme d'ICTAI, il ne peut tirer partie du seuil Q.

8 Propriétés des solutions

Suivant la condition d'arrêt utilisée, on a les propriétés suivantes sur les solutions obtenues.

8.1 Condition d'arrêt : précision ou possibles==valides

solutions : solutions trouvées maximales dans la boite courante (valides $\geq Q$ et valides=possibles) $\cup$ solutions trouvées non prouvées maximales dans la boite courante (petites boites avec possibles $\geq Q$ et valides $\geq Q$ possibles) *cup* solutions potentielles (petites boîtes avec possibles $\geq Q$ et valides $< Q$)

Le posttraitement ne porte que les solutions trouvées. Parmi celles-ci:

- On garde les solutions maximales.
- si 2 solutions partagent la moitié de leurs points , on garde la meilleure (ordre total) critère principal ; nombre d'inliers critère secondaire somme des valeurs absolues de la fonction f (compatibilité des observations).

Propriété des solutions gardées : elles sont maximales parmi les solutions trouvées.

Propriéte garantie des solutions retirées : elles partagent la moitié des points avec une solution gardée. En triant les solutions selon le critère ci-dessus, l'algo de retrait garantit cette propriété.

8.2 Condition d'arrêt : précision ou possibles-valides ¡ q/2

solutions = solutions trouvées ∪ solutions potentielles (petites boites)
∪ solutions potentielles partageant 1/2 des solutions trouvées.

Les solutions trouvées ne comprennent plus les solutions optimales même quand il n'y a pas de petites boites.

Mais on est sûr que pour chaque solution trouvée, les solutions potentielles dans les sous-boites de la boite de la solution trouvée partagent au moins la moitié des points de cette solution.

même post-traitement sur les solutions trouvées.

9 Résultats expérimentaux

9.1 Plans et cercles

On a comparé les résultats obtenus avec les diverses modifications avec ceux d'ICTAI.

Toutes solutions Les tests ont été les suivants :

- ibex : algo ICTAI dans nouvelle version Ibex. Cette nouvelle version est utilisée dans la suite.
- m3.1 : limitation de la recherche de points valides; cette limitation est ensuite appliquée à tous les autres tests.
- m4 : limitation dans les appels à la q-intersection
- m14 : limitation dans les appels à la q-intersection et nouvelle condition d'arrêt.
- m2(10)34 : recherche outliers (seuil 10) , limitation dans les appels à la q-intersection, 1 appel à mini-ransac
- m2(0)34 : recherche outliers (seuil 0) , limitation dans les appels à la q-intersection, 1 appel à mini-ransac
- m2(10)34b:recherche outliers (seuil 10) , qintersection aff uniquement, 1 appel à mini-ransac

Optimization Les expérimentations suivantes ont été réalisées :

- ibex : algo optimisation dans nouvelle version ibex , sans améliorations post-ictai
- m3.1 : limitation calcul points valides
- m3.1m4 : limitation calcul points valides + limitation q-intersection

Table 1: Recherche de solutions : Temps cpu

Test case	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	H_{40}	C_1	C_2
ICTAI	1.1	2.4	3.8	18.3	36.4	107.5	53.7	92.5	221.4	24.5	2.3	6.0
ibex	1.0	2.5	4.1	19.6	38.6	113.6	59.1	103.2	245.2	24.5	2.2	5.9
m3.1	1.0	2.3	3.7	17.5	34.5	102.0	53.4	92.4	221.3	23.1	2.1	5.7
m4	0.8	1.7	2.8	13.5	26.5	79.1	37.9	65.6	156.1	20.8	2.0	5.4
m14	0.7	1.7	2.7	13.5	26.5	79.4	38.0	65.3	157.2	17.9	1.7	5.2
m2(10)34	0.8	1.7	2.7	13.4	26.2	78.0	38.0	65.0	155.0	21.3	2.0	5.4
m2(0)34	0.8	1.7	2.7	13.5	26.7	79.3	38.1	65.5	157.2	20.9	1.6	5.0
m2(10)34b	0.8	1.7	2.7	13.3	26.0	77.3	38.0	65.9	156.7	21.7	2.1	5.7

Table 2: Recherche de solutions : Nombre de noeuds

Test case	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	H_{40}	C_1	C_2
ICTAI	4,786	11,626	20,334	122,590	285,898	1,226,748	61,786	111,558	293,834	646,694	30,344	40,600
ibex	4,786	11,626	20,334	122,558	285,838	1,226,652	61,766	111,528		646,694	26,838	37,096
m3.1	4,786	11,626	20,334	122,590	285,890	1,226,728	61,786	111,558	293,834	646,694	26,838	37,096
m4	5,262	12,038	21,034	134,280	313,314	1,353,368	63,918	115,822	307,124	659,772	27,264	39,366
m14	4,136	11,332	20,376	133,968	312,738	1,352,362	63,652	115,494	306,554	544,412	22,860	34,962
m2(0)34	5,250	11,794	20,606	124,510	283,040	1,136,310	63,062	113,202	296,856	636,978	16518	28306
m2(10)34	5,164	11,136	19,502	109,872	252,276	1,023,700	60,780	106,776	273, 244	630,556	16,304	27,530
m2(10)34b	5,372	11,418	19,906	111,110	254,060	1,027,526	61,680	107,832	274,404	693,372	26,838	37,096

- m3.1m4m2(10): recherche outliers (seuil 10) , limitation dans les appels à la q-intersection et points validess , 1 appel à mini-ransac
- ibexopt bs m3.1 profondeur meilleur fils et toutes améliorations
- ibexopt bs no3.1 profondeur meilleur fils et toutes améliorations sauf limitation points valides
- bfs no3.1 : stratégie meilleur d'abord et toutes améliorations sauf limitation points valides
- bfs m3.1 : stratégie meilleur d'abord et toutes améliorations

Pour les maisons H40 et H44, en optimisation, une précision 0.0001 permet d'obtenir l'optimum sans avoir de petites botes. optimum 38 inliers pour H40 et 171 inliers pour H44 avec une tolérance de 0.001.

problème résolu en 1.2s pour H40 et 166s pour H44 avec ibexopt profondeur (ibexopt bs no3.1)

9.2 Matrice fondamentale

Critère algébrique. Essais avec une boîte initiale autour de la solution 144 (solution Ransac 146). Modélisation à 8 variables (v(3,3)=1), tolérance 0.001, points normalisés.

Table 3: Optimisation : Temps cpu

Test case	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	H_{40}	H_{44}	C_1	C_2
ibex	0.3	0.9	1.1	11.0	16.9	35.8	43.2	65.8	132.2	1.7	454.3	0.9	4.5
m3.1	0.3	0.8	1.0	9.9	15.2	32.4	39.4	60.3	121.4	1.5	412.6	0.8	4.4
m3.1m4	0.3	0.7	0.9	9.0	14.1	30.2	32.8	49.8	100.9	1.3	322.4	0.8	4.4
m3.1 m4 m2(10)	0.2	0.7	0.8	7.7	12.1	26.1	28.5	43.3	88.0	1.1	252.4	0.8	4.4
ibexopt bs no3.1	0.2	0.5	0.7	6.4	9.1	18.1	26.6	39.0	75.5	1.2	142.5	0.7	4.5
ibexopt bs m3.1	0.2	0.5	0.7	6.3	8.8	17.8	26.1	38.6	73.2	1.2	140.6	0.7	4.1
bfs no3.1	0.2	0.5	0.5	7.9	12.0	28.5	29.7	45.3	91.8	1.0	193.9	0.7	4.6
bfs m3.1	0.2	0.4	0.4	7.8	12.0	28.5	29.5	45.9	91.3	1.0	MO	0.7	4.2

Essais suivants réalisés avant la mise en place du polytope intérieur pour la contrainte de déterminant qui semble allonger un peu le temps de calcul (25% sur un exemple)

Par ailleurs, une solution avec Ransac à 153 a été trouvée.

- petites boites
  taille 0.000003 144 prouvé en 16s prec 0.0000005 BF 0.46s
  taille 0.000004 145 prouvé en 87s prec 0.0000005 BF 1.4s NBF 0.44s 1246
  taille 0.000012 146 prouvé en 164s. prec 0.0000005 BF 436s. 60M NBF 166s.
  taille 0.000013 146 prouvé en 536s prec 0.0000002
  taille 0.000015 147 prouvé en 2149s prec 0.0000005
  taille 0.000018 147 prouvé en 84s (91s) prec 0.0000005 NBF 45s 117148 noeuds
  taille 0.00002 147 prouvé en 149s prec 0.0000005 NBF 102s 265984 noeuds
  taille 0.000025 147 prouvé en 691s prec 0.0000005 NBF 711s 1700732 noeuds
  taille 0.00003 146 non prouvé en 10000s. NBF 147 prouvé en 2343s. 5870534 noeuds
  taille 0.00004 146 non prouvé en 10000s.

- taille 0.001 prec 0.001 118 valides 200 possibles 27s
  prec 0.0008 126 valides 197 possibles 53s
  prec 0.0007 131 valides 196 possibles 57s
  prec 0.0006 135 valides 195 possibles 62s
  prec 0.0005 145 valides 187 possibles 1963s
  prec 0.00049 147 valides 185 possibles 2958s
  prec 0.00048 147 valides 185 possibles 2983s
  prec 0.00045 147 valides 185 possibles 3003s
  prec 0.0004 147 valides 185 possibles 3475s
  prec 0.0003 148 valides 185 possibles 4211s
  prec 0.0002 148 valides non prouvé 10000s NBF 152 puis MO

- taille 0.002 prec 0.001 142 valides 210 possibles 2741s.

prec 0.0006 140 valides 198 possibles 4597s.

- taille 0.01 prec 0.01 27 valides 356 possibles 17s
  prec 0.009 40 valides 334 possibles 185s
  prec 0.008 40 valides 334 possibles 195s.
  prec 0.007 46 valides 329 possibles 250s.
  prec 0.006 46 valides 329 possibles 299s.
  prec 0.005 81 valides 314 possibles 5639s.
  prec 0.004 78 valides > 10000s
  prec 0.003 79 valides > 10000s NBF 68 en 2mn , puis 81 en 7m , puis 82
  prec 0.002 108 valides > 10000s

- taille 0.02 prec 0.01 43 valides 364 possibles 7860s.
  prec 0.001 NBF 83 valides MO

- taille 0.05 prec 0.01 22 valides NBF 77 valides MO

9.3 Matrice essentielle

Modélisation 9 variables avec contraintes de matrice essentielle (2 ddl) , de norme (1 ddl) , de déterminant (1 ddl)

9.3.1 Optimisation sous contraintes

Réalisation d'un optimiseur en profondeur d'abord guidé par un oracle. (solution de RANSAC).

- Propagation des contraintes de matrice essentielle : relaxation linéaire (polytope hull) avec 3 contraintes de matrice essentielle sur 9, la norme et le déterminant et linéarisation avec arithmétique affine.
- Recherche de point valide : polytope intérieur coin aléatoire pour chaque contrainte et minimisation aléatoire d'une direction. Au maximum 10 essais par noeud (tant que la zone faisable est non vide).

9.3.2 Q-inter

2 critères : algébrique et Sampson

Critère algébrique Essais réalisés sur exemple test2 et deux réductions 65 et 36 mesures **Exemple réel : test2**

La résolution termine sur l'exemple test2 de usac, avec une tolérance de 0.01, on obtient les résultats suivants : on utilise la projection supplémentaire.

- précision 0.005 [516 520] en 21886s.
- précision 0.01 [516 524] en 14026s.
- précision 0.02 [514 528] en 11390s.

avec hc4 sur les contraintes , y compris les contraintes redondantes, la bissection smearsumrel, on obtient les résultats suivants

- précision 0.001 [516 517] en 46500s
- précision 0.0025 [516 518] en 26352s
- précision 0.005 [516 520] en 17580s.
- précision 0.01 [516 525] en 10022s.
- précision 0.01 1 branche prof 48 [515 530] en 9231s.
- précision 0.01 1 branche prof 48 [515 537] en 6968s. 3930566 branches
- précision 0.01 1 branche prof 40 [514 544] en 4630s. 2643136 branches
- précision 0.01 1 branche prof 40 [513 521] en 5689s. 2982596 branches critre valides-pt milieu
- précision 0.02 [514 528] en 11390s.

  borne sup fausse
- précision 0.0005 [515 516] en 5158s. 2626202 branches prof 40 critre valides-pt milieu
- précision 0.001 [515 516] en 5560s. 2619290 branches prof 40 critre valides-pt milieu
- précision 0.002 [514 515] en 5504s. 2790810 branches prof 40 critre valides-pt milieu
- précision 0.0002 [514 515] en 5473s. 2795200 branches prof 40 critre valides-pt milieu

avec une precision de 5 sur l'objectif

- précision 0.005 [515 521] en 5768s.
- précision 0.01 [515 524] en 4567s.
- précision 0.01 1 branche prof 40 [512 544] en 3633s.
- précision 0.01 critère milieu 1 branche prof 40 [512 521] en 4412s et 2324468 branches
- précision 0.02 [512 528] en 6786s.

estimessentialmatnorm essentialtest2.txt 10 0.001 2 0.015 1.e-4 0 0 100000 2 avec une tolérance 0.001, c'est plus long.

Exemple 65 mesures

Tous essais avec précision de 1 sur objectif.

- avec tolérance 0.01 , précision de 0.0001
  - algo complet solution 58 petites boites 59 pas termin en 10000s. [58 65]
  - algo 1 branche profondeur 55 solution 60 petites boites 61 en 955s. [60 61]
  - algo 1 branche profondeur 50 solution 60 petites boites 61 en 760s. [60 61]
- avec tolérance 0.01 , précision de 0.00005
- algo 1 branche profondeur 50 solution 60 petites boites 61 en 659s [60 61]

Exemple 36 mesures

Tous essais avec précision de 1 sur objectif.

- avec tolérance 0.01 , précision de 0.0001
  - algo complet : solution 34 prouvée en 1060s.
  - algo 1 branche profondeur 50 solution 34 prouvée en 244s. [34 34]
  - algo 1 branche profondeur 45 solution 34 non prouvée en 261s. petites boites 35 [34 35]
- avec tolérance 0.005 et précision 0.00005
  - algo complet : solution 33 prouvée en 437s. [33 33]
  - algo 1 branche profondeur 50 solution 33 prouvée en 335s. [33 33]
  - algo 1 branche profondeur 45 solution 33 non prouvée en 331s. petites botes 34 [33 34]
- avec tolérance 0.002 et précision 0.0001
  - algo complet solution 30 non prouvée . pas terminé en 10000s.
  - algo 1 branche profondeur 50 solution 33 prouvée en 588s.
- avec tolérance 0.001 et précision 0.005
  - algo 1 branche profondeur 50 solution 29 non prouvée en 16231s. petites boites 34. précision 0.005 [29 34]
  - algo 1 branche profondeur 55 solution 30 non prouvée en 17474s. petites boites 34. précision 0.0001 [30 33]

  - algo 1 branche profondeur 60 solution 30 non prouvée en 34248s. petites boites 33. précision 0.0001 [30 33]

- avec tolérance 0.0005 et précision 0.005
  - algo 1 branche profondeur 50 solution 21 petites boites 33 algo non terminé en 10000s. [21 36]

Critère de Sampson Pour Sampson, mise en place sur exemple test2 de usac (582 correspondances), on trouve une solution autour de 424 - 431 inliers avec une tolérance de 1.

plus ou moins rapidement selon l'oracle de départ (solution RANSAC)

avec oracle 279 inliers, on trouve rapidement (4mn) une solution à 330-340 inliers , puis la solution à 420 - 430 inliers en 50000s avec précision 0.01

Mise en place d'un seuil sur la q-inter et fwdbwd avant seuil (> 0.1) fwd uniquement , après seuil fwdbwd + q-inter mêmes résultats : solution 424 en 12h.

Mise en place de la projection affine : cela ne semble pas rentable (coût trop important de l'arithmétique affine) : 316 en 20mn, pas mieux en 12h. seuil à envisager ??

avec prec0.02 solution 388 en 12h.

Exemple 36 mesures

sur petit problme (36 mesures), la projection affine semble rentable : solution 29 trouvée avec epsobj5 avec domaine total en 3600s.

(pb fermé en [28 36] avec epsobj6 sans proj affine en 14689s.) pb fermé en [30 34] avec epsobj3 prec 0.005 , limite bissection profondeur 60, avec proj affine en 48595s.

exemples avec tolérance 1 précision 0.005

- précision objectif 3
  - algo complet [30 33] en 51514s.
  - algo 1 profondeur 65 [30 33] 51516s.
  - algo 1 profondeur 60 [30 34] 51212s.
  - algo 1 profondeur 58 [30 34] 46638s.
  - algo 1 profondeur 55 [29 34] 78393s.

- précision objectif 2
  - algo complet précision 0.01 [29 33] 77102s.
  - algo 1 profondeur 65 [31 33] 59780s.

- précision objectif 1
  - algo complet précision 0.001 ne termine pas en 100000s. solution 31 petites boites 32 [31 36]
  - algo 1 profondeur 65 ne termine pas en 100000s. solution 31 petites boites 33 [31 36]